Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 2, February 2014, pg.290 – 301

RESEARCH ARTICLE

APPLICATION OF CLOUD COMPUTING AS HPC PLATFORM FOR EMBEDDED SYSTEMS

N.S.Gawai[1], B.V.Chikte[2], V.R.Pandit[3], S.M.Ingale[4], M.S. Burle[5]

[1] Electronics & Telecommunication Engg., SGB Amravati University, J.D.I.E.T, Yavatmal, India
[2] Electronics & Telecommunication Engg., SGB Amravati University, J.D.I.E.T, Yavatmal, India
[3] Electronics & Telecommunication Engg., SGB Amravati University, J.D.I.E.T, Yavatmal, India
[4] Electronics & Telecommunication Engg., SGB Amravati University, J.D.I.E.T, Yavatmal, India
[5] Electronics & Telecommunication Engg., SGB Amravati University, J.D.I.E.T, Yavatmal, India
[1] nitingawai@rediffmail.com; [2] bablu_chikte@yahoo.com;
[3] vaibhavpandit111@gmail.com; [4] saurabh2010electrical@gmail.com; [5] milindburle80@gmail.com

Abstract— ***The process for prototype of a High Performance Computing platform for robotics is discussed in this paper. The Cloud-based platform gives advantages of infinite resources, dynamic scalability and better resource utilisation. The different elements and technologies involved in a Cloud will be presented, as well as explaining the building process. The Cloud is compared with using Open stack middleware and KVM with hardware-based virtualization for performance.***

Keywords— *Eucalyptus; HPC; Virtualization technology; KVM; Cloud Controller*

I. INTRODUCTION

Traditionally, information processing tasks were performed in centralized systems or independent computers. Nevertheless, some of them depended on mobile elements, such as navigation in robots. These kind of operations were done by the robot's embedded hardware. However, we cannot expect to get good performance from some high performance tasks such as face recognition, surveillance, voice processing, natural language processing. etc, as they consist on very complex algorithms. In addition to this, we must take into account other facts such as power consumption, which is vital for the efficiency of such systems. Equipping a robot with hardware that we might find in a desktop PC has this trade-off, resulting in an useless and expensive machinery. Because of this, the question that we want to answer is: could we use an external auxiliary platform that performs all those computational hard operations? Thank to this, provided we have an efficient communication system, the robot would obtain the results faster, with the extra benefit of consuming less power. This can be especially interesting for situations where we need as least response time as possible. In order to give an answer to this question, this paper discusses the building of a first prototype, which follows the Cloud Computing approach. Cloud Systems are becoming increasingly popular during the last years, for the different characteristics it offers, such as infinite resources and dynamic elasticity. Large enterprises such as Amazon or Microsoft are focusing in cloud services as a interesting area of business. For example, small business which require to perform a computationally hard task, can use the resources offered by a cloud for a relatively small fee rather than spending a big capital to get all the bare metal resources necessary to get the computation done (this becomes even more useful if these computations are not regular). Another example of cloud services are those provided by Dropbox, that is, using the cloud as a storage resource. The cloud can also provide a framework to develop applications which exploit its characteristics, such as Windows Azure or Google App Engine. Therefore, the possibilities that the cloud can offer to users and developers are vast. Figure 1 shows the concept of the prototype built, using as an example a senior person. The following sections will cover the different aspects that must be taken into account when building this kind of systems.

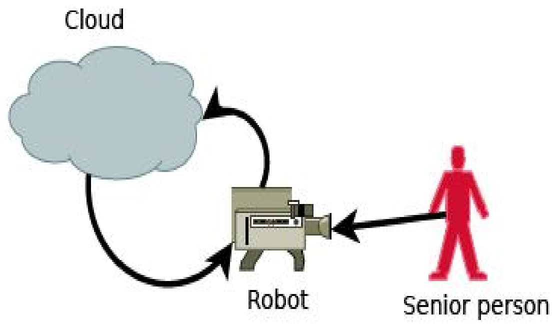

Fig.1 Platform utilization example

1) To Get Information
2) Get the information processed by the Cloud
3) Act accordingly

II. DESCRIPTION OF CLOUD COMPUTING

Cloud computing is a relatively new discipline in large scale distributed computing. It is based on the idea of utilising computational resource in the internet for a certain purpose (Dikaiakos et al., 2009). However, this concept can be applied to other large scale computing areas such as Grid Computing. In that sense, there is a lack of coherence between the different definitions of Cloud Computing, and what makes it different from other platforms such as the Grid. For instance, in (Laszewski et al.,), it is defined as a reliable, customized and QoS guaranteed dynamic computing environments provider, whereas in (Foster et al., 2008) it states that it is based on a the delivery of a pool of abstracted, virtualized, dynamically scalable, managed computing power, storage, platforms, and services. The most interesting element of these definitions are the dynamically-scalable and virtualized nature of the resources provided by the Cloud. About the first term, in the context of High Performance Computing, for instance, it can be quite useful the possibility of adding (or removing) new computing nodes at runtime when needed. Therefore, new load balancing mechanisms can be studied (as most of the known mechanisms are based on systems where the number of computing nodes does not vary). About the second term, the main implication is that users will not access the bare metal resources in the cloud. For an HPC system this fact can be especially significant when measuring the performance for HPC applications, as we may obtain different results depending of where the virtual resources are located. Moreover, we can also move virtual resources from one bare metal node to another without significant loss of functionality or performance, being our whole system more maintainable. Figure 2 shows the structure of a cloud system in layers. The deepest level that a user can access is that of virtual machines. As we can see, depending on the depth of access, we can have different types of services:

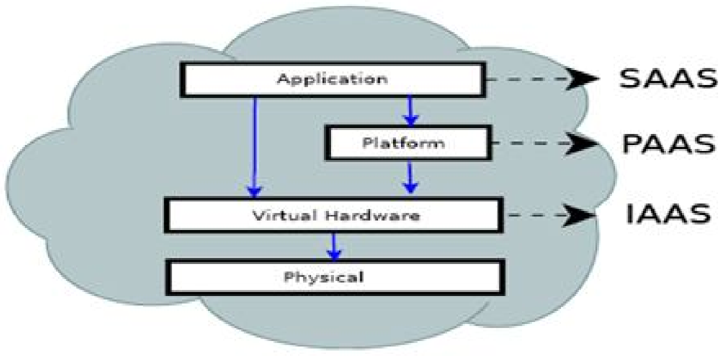

Fig.2 A cloud system structured in layers

Software as a Service: In this category the consumer uses end-user applications that are provided by the cloud (Application Layer). One example is that of Google Mail.

Platform as a Service: In this category the user can develop applications for the cloud by using a particular framework (Platform Layer), where some processes such as the Virtual Machine Creation or Load Balancing are already implemented in the given API.

Infrastructure as a Service: In this kind of service the user has access to the virtual machines directly (Unified Resource Layer). The advantage is that the user has more freedom for developing but, at the same time, more work to do as there is not any API that helps the user with certain processes.

III. RELATED WORK IN CLOUD COMPUTING

The on-demand, pay-as-yougo model of Cloud Computing creates a flexible and cost-effective means to access compute resources. For these reasons, the scientific computing community has shown increasing interest in exploring cloud computing (Jackson et al., 2010). Experiments done show that the performance obtained shows severe performance issues in network performance due to the virtual layers that our HPC Job must deal with (Salmeron, 2011; Jackson et al., 2010), with some solutions proposed in (Xiaoyong, 2011) in the specific case of finite-element processing. To make matters worse, some strange fluctuations in performance on MPI Jobs have been reported (El- Khamra et al., 2010). However, these experiments do not use hardware-assisted virtualisation (explained in section 4.3), and there are also some positive experiments in the case of agent-based simulations, with proper speedup obtained (Decraene et al., 2010). Therefore, we can conclude that there is still some debate about the efficiency of the cloud for HPC jobs. More experiments have been done with different virtualisation platforms, which is directly related with Cloud Computing (Regola and Ducom, 2010; Walters et al., 2008; Che et al., 2010). These results will be explained in more detail in section 4.3. _ Cloud Computing for Robotics: This area is becoming of high interest judging the number of papers appeared in the recent years. In (Agostinho et al., 2011) a cloud platform for robotics workflows using the Platform as Service approach is developed, concluding that it offers a networked robotics platform as a service with strong advantages in performance without compromising security. In (Chen et al., 2010) the paradigm Robot as a Service (RaaS) is presented, proposing a cloud framework for interacting with robots in the area of service oriented computing. In terms of HPC, in (Arumugam et al., 2010) a platform for solving parallelizing the FastSLAM algorithm in Map/Reduce is built, exploring the possibilities of creating a global map between several robots, thanks to the Cloud. As we can see, the idea of using a cloud platform for robotics has become more than reasonable. Basic ideas and challenges are explained in (Hu et al., 2012). Another area of interest is that of elder people, in which the GameUP project (ATC department) is mainly focused on. The use of Cloud Computing for senior people companion (for tasks such as speech/sound recognition, speaker identification, face identification, sound source estimation and text to speech) is presented in (Chen et al., 2011) and for healthcare in (Abidi, 2011).

IV. BUILDING THE PLATFORM

This section focuses on preparing the environment, which is separated in different concerns: select the cloud middleware, decide the virtualization technology and using an HPC algorithm to verify it.

A. Bare metal nodes specifications

The bare metal resources available may consist on few 64-bit computers with quadcore having Gigabit Ethernet Connection with Intel VT Extensions for Hardware Based Virtualisation. Figure 3 shows the network interconnection of the computers, which are located in Lab F0.30 in the School of Computer Engineering at Seville University. Due to the School of Computing Engineering IT Management Centre restrictions, we only have a partition of 10 GB, which will suppose a Cloud with little storage possibilities, but more than enough to verify the platform.

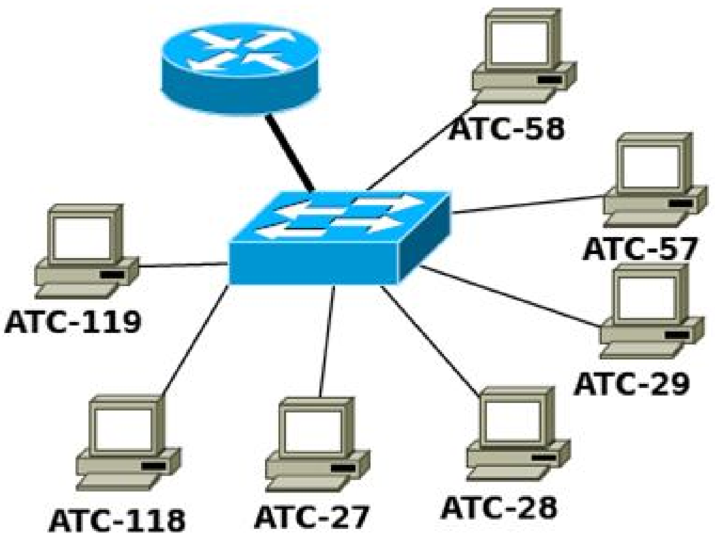

Fig.3 Bare Metal Cluster

B. Choosing The Cloud Middleware

The first step is to decide which Cloud middleware will be used to create the Cloud. In order to do that, we have investigated two strong open-source cloud technologies in the market: Eucalyptus and Openstack.

1) Description of Eucalyptus

Eucalyptus is an open-source Cloud Middleware developed by Eucalyptus Systems, Inc. written in Java and C. It is designed to be simple, flexible and modular with a hierarchical design (Nurmi et al., 2009) This middleware's architecture is "simple, flexible and modular with a hierarchical design". This architecture can be seen in figure 4. The main elements are the Node Controller, responsible of managing the Virtual Machines in a certain host, the Cluster Controller, which manages and schedules a set of node controllers forming a cluster and a virtual instance network, an Storage Controller, responsible of storing and allowing access to virtual images and user data, and a Cloud Controller, which gathers information about all the virtual resources by using the cluster controllers and performs high level scheduling. Eucalyptus allows two different virtualization technologies to be used: KVM and Xen.

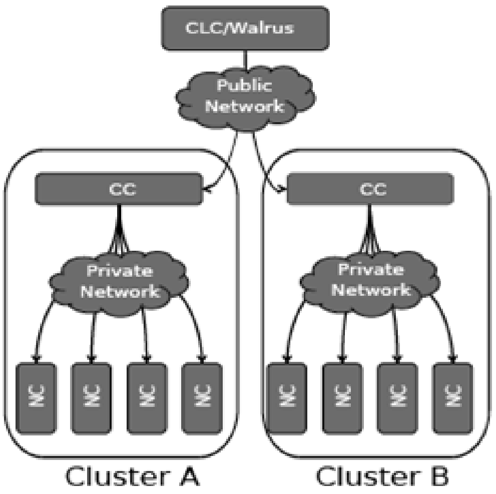

Fig.4 Eucalyptus Architecture, extracted from (Nurmi et al., 2009)

2) Description of Openstack

Openstack is a set of services created by Openstack Foundation, founded by Rackspace Hosting and NASA, in order to fight Amazon's market leadership, which does not allow easy migration to other platforms (Alacot Torres, 2011). Openstack project is the one that has support from different companies, such as Dell, Intel, AMD, Cisco and Canonical. The services that can be installed in the different nodes are the following (figure 5 shows the architecture):
 Nova: a Management Platform that manages compute resources, networking, authorization, and scalability needs of the cloud.
Glance: OpenStack Imaging Service is a lookup and retrieval system for virtual machine images. It can be configured to use different storage backends such as Amazon S3, Openstack Object Store or HTTP.
Swift: provides a distributed, eventually consistent virtual object store. It is capable of storing billions of objects distributed across nodes.
 Keystone: provides identity and access policy services for all components including (but not limited to) Swift, Glance, Nova.
 Horizon: web based dashboard that can be used to manage OpenStack services Just like Eucalyptus, Openstack also supports KVM and Xen.

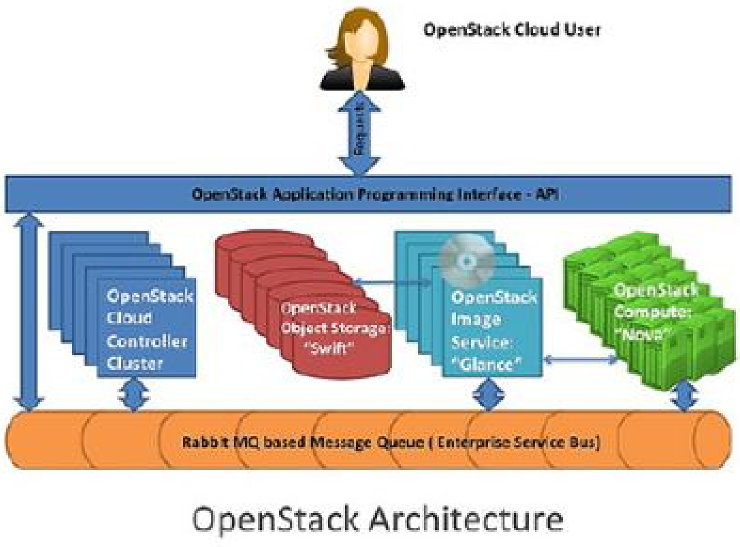
OpenStack Cloud User
Requests
OpenStack Application Programming Interface - API
OpenStack Cloud Controller Cluster
OpenStack Object Storage: "Swift"
OpenStack Image Service: "Glance"
OpenStack Compute: "Nova"
Rabbit MQ based Message Queue (Enterprise Service Bus)
OpenStack Architecture

Fig.5 Openstack Architecture

3) *Final Choice*

As seen in both descriptions, both technologies have similar features, and both have good community support. Untill now, Eucalyptus 3.0 is in alpha stage and version 2.0.3 is the latest stable version. This version was experimented in (Salmeron, 2011) and several bugs were detected. In addition to this, Ubuntu has default support for Openstack (as Canonical is one of the main supporters), making the Cloud installation process and configuration easier. Therefore, the choice of Openstack as a Cloud Middleware is obvious. Moreover, Openstack has a better web frontend (Horizon).

C. Choosing the Virtualization Technology

After having the cloud middleware, the next step is to decide which virtualisation technology is more suitable for the cloud environment. As stated before, the two possibilites are KVM and Xen. It is necessary to choose which technology is more suitable for the cloud for implementation. The idea of computer virtualisation is that a layer is added between the hardware and operating system. This is a virtualisation layer, and allows multiple operating system instances (also know as Guest Operating Systems) to run at the same time within virtual machines on a single computer, dynamically partitioning and sharing the available physical resources such as CPU, storage, memory and I/O devices (VMWare, 2007). Figure 6 shows the architecture.

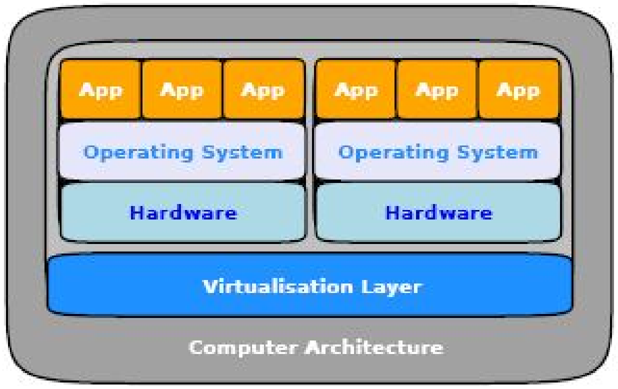

Fig.6 General Visualization Layer

The problem of virtualisation is that the Guest OS needs to access the hardware directly in order to execute a set of privileged instructions. Therefore the virtualisation layer must trap and translate at run time these sensitive instructions in order to allow virtualisation. Figure 7 shows an example of different privilege levels in a x86 system. The lowest level implies access to the hardware layer (cannot be directly executed by the VM). This example will be used in subsection 4.3.1.

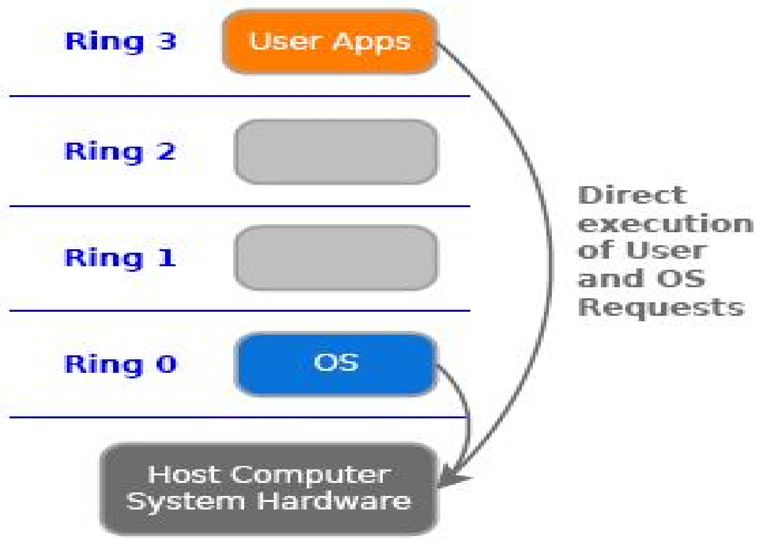
Ring 3
User Apps
Ring 2
Ring 1
Ring 0
OS
Host Computer System Hardware
Direct execution of User and OS Requests

Fig.7 Privilege levels in a x86 system

1) *Types of Virtualization*

We now discuss common virtualization implementations, outlining the advantages and disadvantages, and thus being able to understand what each virtualisation technology can offer. The main difference between the different approaches is how the sensitive instructions that the Guest OS wants to execute are handled:

Full Virtualization: This paradigm is based on the inclusion of a Virtual Machine Monitor (VMM) or hypervisor in the Virtualisation layer. This translates kernel code to replace non-virtualizable instructions with new sequences of instructions that will do the same effect on the virtualized hardware (VMWare, 2007). The Guest Operating system will not be aware that it is being executed in a virtual machine, having then a total decoupling between the hardware layer the virtual resource and thus no other assistance (via hardware or via the Host OS) is necessary. Figure 8 shows the architecture of the virutalisation in a x86 computer. The advantage of this approach is the big compatibility with the available operating systems, as it does not have to be modified in order to be virtualized.

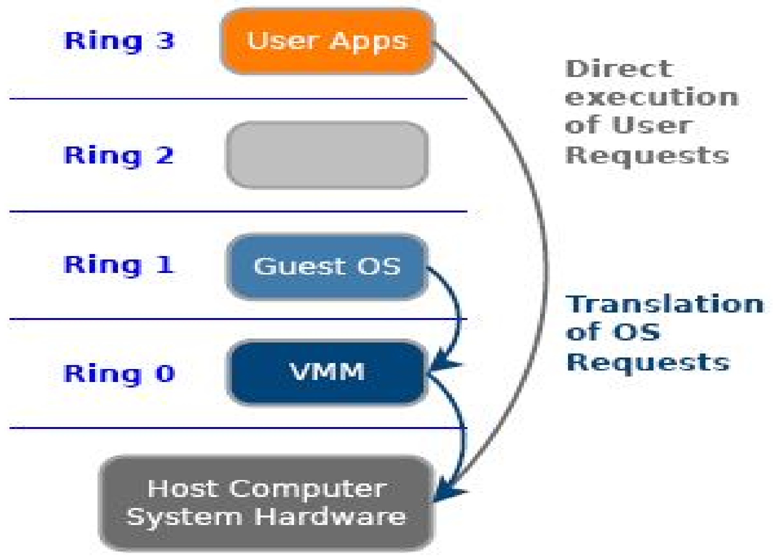
Ring 3
User Apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host Computer System Hardware
Direct execution of User Requests
Translation of OS Requests

Fig.8 Full Virtualisation architecture in a x86 machine

Paravirtualisation: This approach is also based on the idea of hypervisor in the virtualization layer, but in this case the guest operating system kernel is modified. The modification consists on substituting the sensitive instructions with hypercalls to the virtual machine manager, which will execute the necessary instructions that will perform the desired effect on the Guest OS. Figure 9 shows the architecture in a x86 computer. The use of this modified kernel makes the translation process faster, obtaining then a boost in performance if we have to use privileged instructions. This can be the main advantage of this virtualization approach. On the other hand, in this case the operating system is not completely isolated from the rest of the layers, as it knows that it is being virtualized.

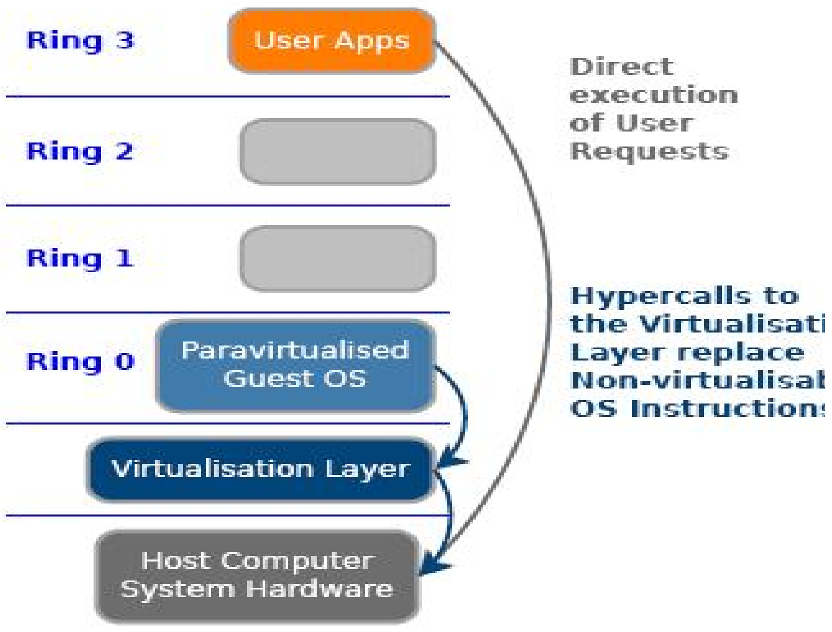
Ring 3
User Apps
Direct execution of User Requests
Ring 2
Ring 1
Hypercalls to the Virtualisati Layer replace Non-virtualisal OS Instruction:
Ring 0
Paravirtualised Guest OS
Virtualisation Layer
Host Computer System Hardware

Fig.9 Paravirtualisation architecture in a x86 machine

Hardware Assisted Virtualisation: The new generation of CPU's are including new features that make the virtualization process easier. The main idea is to remove the need of translating the privileged instructions in order to be executed. In order to achieve that, a new execution mode may be where the privilege instructions can be executed. This is where the hypervisor will be running, and the trapping of sensitive instructions is done automatically without any type of further translation. Figure 10 shows how the resulting architecture for x86 computers. This new approach may give better performance than the previous two options. Actually, this approach is combined with the other two paradigms.

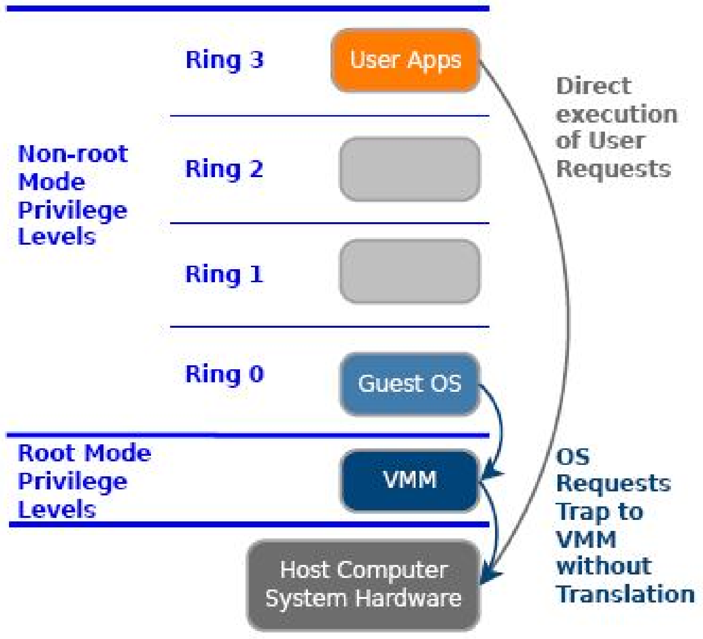
Ring 3
User Apps
Direct execution of User Requests
Non-root Mode Privilege Levels
Ring 2
Ring 1
Ring 0
Guest OS
Root Mode Privilege Levels
VMM
OS Requests Trap to VMM without Translation
Host Computer System Hardware

Fig.10 Hardware based virtualisation architecture in an x86 machine

2) *Description of KVM*

KVM (Kernel-based Virtual Machine) is a Linux open-source virtualisation technology based on the Full Virtualisation paradigm (Walters et al., 2008). Figure 11 shows KVM's architecture. IT converts the linux kernel into a hypervisor by loading a module, which will add a guest mode to it and export a device. This module will allow each virtual machine to see a separate address space, and therefore isolation of the guests is guaranteed. Finally, for the I/O virtualization QEMU is used. In newer versions, KVM requires hardware assisted virtualization in order to run properly, therefore we can have the benefits of full operating system virtualization and the reduction of overheads obtained with hardware based virtualization. Because it follows the full virtualization paradigm, the main advantage of KVM is that it allows any operating system to be run (the only exception is those of different architectures from the bare metal node), allowing the users to have full freedom to create any type of virtual machine as per requirement.

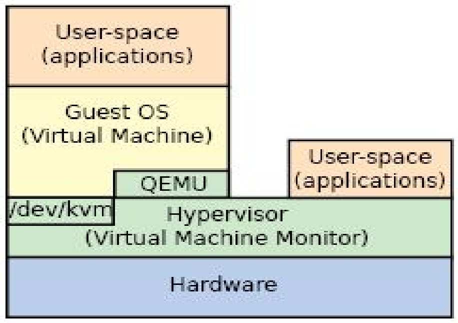
User-space (applications)
Guest OS (Virtual Machine)
QEMU
User-space (applications)
/dev/kvm
Hypervisor (Virtual Machine Monitor)
Hardware

Fig.11 KVM's Architecture

3) *Description of Xen*

Xen is another open-source virtualization platform, but this time it is based on the paravirtualisation paradigm (though it also supports full virtualisation). Figure 12 shows the architecture of this virtualization solution. The main elements are (Xen, 2008):

The Xen Hypervisor: It controls the execution of the virtual machines and abstracts the hardware layer for the virtual machines. It has no responsibilities about the I/O devices residing in the computer system.

Domain 0: This is a virtual machine that has access to all the physical resources existing on the machine. It interacts with the rest of the running virtual machines, and will process all the I/O requirements they may have.

Domain U: This represents a running virtual machine. Depending on the nature of the virtualization we have two types of Domain U: Paravirtualised (PV) or Full Virtualised (HVM).

Domain Management and Control: This represents a set of services that allow the virtualization process management.

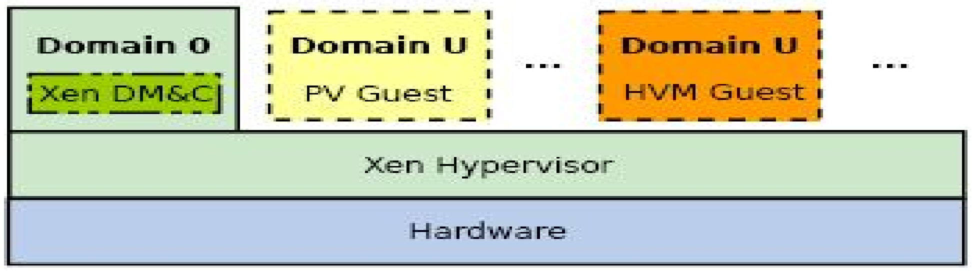

Fig.12 Xen Architecture

4) Final Choice

As the main aim of our cloud is to be a HPC platform, we must choose the virtualization platform that gives the best performance. In that sense, we analysed different benchmark reports in order to make a decision. In (Che et al., 2010) we can see that, after doing tests with SPEC CPU2006, LINPACK, Kernel compiling, RAMSPEED, LMbench, IOzone, Bonnie++, NetIO, WebBench, SysBench and SPEC JBB2005, they state that Xen has better performance than KVM (they also analyse OpenVZ but, as it is not supported by Openstack, we ignored these results). In (Regola and Ducom, 2010) KVM and Xen are again compared in the context of High Performance Computing in different areas: Disk usage, Network performance and Parallel Applications. Once again, the result is that Xen has a slightly better performance. Similar tests are done in (Walters et al., 2008), showing that Xen should be able to handle HPC jobs as well (KVM is not compared in this literature reference). However, these performance benchmarks do not seem to use hardware assisted virtualisation, therefore they are outdated. With the hardware available, it would be more of interest testing how the overheads improvements of Intel VT Extensions affect the overall performance. Because of this, we chose KVM as it is prepared for this kind of virtualization technology.

D. Cloud Configuration

After selecting the cloud middleware, virtualization platform and knowing the hardware, we must select a cloud layout. We chose the simplest layout available, which consists of (as seen on figure 13):

A cloud controller, which has the Glance, Horizon, Keystone and all Nova services but nova-compute, responsible of running virtual instances.

Compute nodes, which only have the nova-compute service

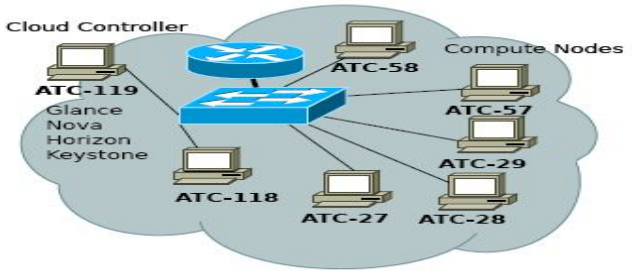

Fig.13 Cloud layout

E. Test with a typical HPC problem

Once our cloud is built, we can validate it as a HPC platform. In order to do so, we will use as a example a typical HPC Problem: a Laplace's equation solver:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \qquad (1)$$

A finite difference approach will be used. The method use is that of Gauss-Seidel iterative method with red-black ordering:

$$\phi_{i,j}^{k+1} = \begin{cases} \frac{1}{4}\left(\phi_{i+1,j}^{k} + \phi_{i-1,j}^{k} + \phi_{i,j+1}^{k} + \phi_{i,j-1}^{k}\right) & (red) \\ \frac{1}{4}\left(\phi_{i+1,j}^{k+1} + \phi_{i-1,j}^{k+1} + \phi_{i,j+1}^{k+1} + \phi_{i,j-1}^{k+1}\right) & (black) \end{cases} \quad (2)$$

This problem is suitable for testing because it involves not only computation but synchronization between instances. We can obtain better performance with our cloud with several VM in each one. The reason of this is because the code used is not prepared to be executed in multiple cores. Therefore with the bare metal nodes we cannot take advantage of the 4 cores. Further each VM may be executed in single core, and because of this we have better resource utilisation. Hence, we can assure that Cloud Computing can perform better with single core algorithms with several virtual machines in each compute node.

V. CONCLUSIONS

This paper is an attempt to establish the efficiency of a Cloud Computing environment for HPC Jobs and also a good support platform for an embedded system. However, this process was not trivial. First of all the Cloud Middleware and Virtualisation Technology had to be chosen. We saw that both open-source leading technologies (Eucalyptus and Openstack), despite its modularity and ease of installation, need some time to get configured properly.

Moreover, knowledge of the different virtualization technologies and paradigms is needed. The election of the proper paradigm depends widely on the available hardware. This decision can affect severely in the final result, as seen on (Salmeron, 2011) and (El-Khamra et al., 2010), as not every piece of hardware might be suitable for a Cloud Environment. Regarding this matter, we have seen the efficacy of Hardware-based Virtualization (Intel Virtualization Extensions), obtaining performance results near to the Bare Metal Resources containing the Virtual resources.

We can be sure that a Cloud can potentially be a very interesting platform for embedded applications. We have set the base for a future intensive research for not only obtaining better performance, but also better resource management and power consumption, other important aspects in embedded systems in general.

REFERENCES

[1] Abidi, F. (2011). Cloud computing and its effects on healthcare, robotics, and piracy. In Sustainable Technologies (WCST), 2011 World Congress on, pages 135 –140.

[2] Agostinho, L., Olivi, L., Feliciano, G., Paolieri, F., Rodrigues, D., Cardozo, E., and Guimaraes, E. (2011).

[3] A cloud computing environment for supporting networked robotics applications. In Dependable, Autonomic and Secure Computing (DASC), 2011 IEEES.

[4] Dikaiakos, M. D., Katsaros, D., Mehra, P., Pallis, G., and Vakali, A. (2009). Cloud computing: Distributed internet computing for IT and scientific research. IEEE Internet Computing, 13(5):10–13.

[5] Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., Fu, C., and Wang, L. Cloud computing: a perspective study. New Generation Computing, 28(2):137–146.

[6] Chen, Y., Du, Z., and Garci anda Acosta, M. (2010). Robot as a service in cloud computing. In Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on, pages 151 –158.

[7] Decraene, J., Cheng, Y. Y., Low, M. Y., Zhou, S., Cai, W., and Choo, C. S. (2010). Evolving agent-based simulations in the clouds. pages 244–249

[8] Chen, Y.-Y., Wang, J.-F., Lin, P.-C., Shih, P.-Y., Tsai, H.- C., and Kwan, D.-Y. (2011). Human-robot interaction based on cloud computing infrastructure for senior companion. In TENCON 2011 - 2011 IEEE Region 10 Conference, pages 1431 –1434.

[9] Ninth International Conference on, pages 1110 – 1116.Alacot Torres, M. (2011). Implantación de una plataforma de cloud computing. Master's thesis, Universitat Politecnica de Valencia.

[10] A synthetical performance evaluation of OpenVZ, xen and KVM. In 2010 IEEE Asia-Pacific Services Computing Conference, pages 587–594, Hangzhou, China.

[11] El-Khamra, Y., Kim, H., Jha, S., and Parashar, M. (2010). Exploring the performance fluctuations of HPC workloads on clouds. In 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pages 383–387.

[12] IEEE. Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-Degree compared. In Grid Computing Environments Workshop, 2008. GCE '08, pages 1–10.

[13] Xen (2008). Xen architecture overview. Technical report, Xen. Xiaoyong, B. (2011). High performance computing for finite element in cloud. In 2011 International Conference on Future Computer Sciences and Application (ICFCSA), pages 51–53. IEEE.

[14] S. (2008). A comparison of virtualization technologies for HPC. In 22nd International Conference on Advanced Information Networking and Applications (aina 2008), pages 861–868, Gino-wan, Okinawa, Japan.

[15] VMWare (2007). Understanding full virtualization, paravirtualization, and hardware assist. Technical report, VMware. Walters, J. P., Chaudhary, V., Cha, M., Jr., S. G., and Gallo,

[16] IEEE Computer Society. ACM ID: 1577895. Regola, N. and Ducom, J. (2010). Recommendations for virtualization technologies in high performance computing in 2010 IEEE Second International Conference on Cloud Computing Technology and Science, pages 409–416, Indianapolis, IN, USA.

[17] Davinci: A cloud computing framework for service robots. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 3084 –3089. Che, J., Shi, C., Yu, Y., and Lin, W. (2010).

[18] IEEE.Hu, G., Tay, W. P., and Wen, Y. (2012). Cloud robotics: architecture, challenges and applications. Network, IEEE, 26(3):21 – 28.

[19] Jackson, K., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H., and Wright, N.(2010). Performance analysis of high performance computing applications on the amazon web services cloud. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, pages 159 –168.

[20] Arumugam, R., Enti, V., Bingbing, L., Xiaojun, W., Baskaran, K., Kong, F. F., Kumar, A., Meng, K. D., and Kit, G. W. (2010).

[21] Salmeron, J. (2011). Implementing eucalyptus and running an agent-based simulation in the cloud. Master’s thesis, School of Engineering, Cranfield University.

[22] Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., and Zagorodnov, D. (2009).

[23] The eucalyptus Open-Source Cloud-Computing system. In Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID ’09, page 124–131, Washington, DC,USA.